



# Structured Assurance Case Metamodel (SACM)

Version 2.2 ~~3~~

---

OMG Document Number ~~formal/21-11-01 [smse/21-11-01]~~

Release Date TBD

Normative Reference: <https://www.omg.org/spec/SACM/2.2/PDF> ~~3~~

Associated Normative Machine Consumable Files:

<https://www.omg.org/spec/SACM/20190235/emof.xml>

---

 2022  
Copyright © 2010-~~2021~~, The MITRE Corporation  
Copyright © 2010-2019, Adelard LLP  
Copyright © 2010-~~2021~~, The University of York  
Copyright © 2015-~~2021~~, Universidad Carlos III de Madrid  
Copyright © 2015-~~2021~~, Carnegie Mellon University  
Copyright © 2010-2019, Benchmark Consulting  
Copyright © 2010, Computer Sciences Corporation  
Copyright © 2010-~~2021~~, KDM Analytics, Inc.  
Copyright © 2010-~~2021~~, Lockheed Martin  
Copyright © 2017-~~2019~~, Elparazim - 2021  
Copyright © 2021, Northrop Grumman - 2021  
Copyright © 2021, Dalian University of Technology  
Copyright © 2017-~~2021~~, Object Management Group, Inc.  
Copyright © 2022, Elparazim  
Copyright © 2022, Fraunhofer Gesellschaft

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

#### LICENSES


The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

#### PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

In SACM 2.2, minor clarifications, expanded explanations for enhanced readability and consistency have been incorporated along with a new annex to provide additional details about the concepts of package interfaces and bindings and their use within the standard.

 In SACM 2.3, a new annex to provide a UML profile for to allow UML-based tools to work with SACM concepts and the allowing for categories to contain categories as a mechanism for organizing concepts in the Terminology class.

## 2 Conformance

### 2.1 Introduction

The Structured Assurance Case Metamodel (SACM) specification defines the following three compliance points:

- Argumentation Model
- Artifact Model
- Assurance Case Model
- Terminology Model

### 2.2 Argumentation Model compliance point

Software that conforms to the SACM specification at the Argumentation Model compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in the Argumentation subpackage of the SACM specification, including the common elements defined in the Common and Predefined diagrams of the SACM. The top object of the Argumentation package as a unit of interchange shall be the `Argumentation::ArgumentPackage` element of the SACM.

Conformance to the Argumentation Model compliance point does not entail support for the Evidence subpackage of SACM, or the terminology sub package of the SACM.

This compliance point facilitates interchange of the structured argumentation documents produced by existing tools supporting existing structured argument notations such as the Goal Structuring Notation (GSN) and the Claims-Arguments-Evidence (CAE) notation which provide their own mapping onto SACM argumentation aspects. Further details of these mappings are given in Annex A.

### 2.3 Artifact Model compliance point

Software that conforms to the specification at the Artifact Model compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in this Artifact subpackage of the SACM specification, including the common elements defined in the Common and Predefined diagrams of the SACM. The top object of the Evidence package as a unit of interchange shall be the `Artifact::ArtifactPackage` element of the SACM.

Conformance to the Artifact Model compliance point does not entail support for the Argumentation subpackage of SACM, or the terminology diagram of the SACM. This compliance point facilitates interchange of the packages of evidence. In particular, this compliance point facilitates development of evidence repositories in support of software assurance and regulatory compliance.

### 2.4 Assurance Case Model compliance point

This compliance point is mandatory. Software that conforms to the specification at the Assurance Case Model compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in this entire specification. The top object of the Assurance Case package as a unit of interchange shall be the `SACM::AssuranceCasePackage` element.

The Conformance clause identifies which clauses of the specification are mandatory (or conditionally mandatory) and which are optional in order for an implementation to claim conformance to the specification.

## 6 Additional Information

### 6.1 Changes to Adopted OMG Specifications [optional]

This specification completely replaces the SACM 2.1 specification.

### 6.2 Acknowledgements

The following companies submitted this specification:

- MITRE Corporation
  - Adelard LLP
  - KDM Analytics
  - Lockheed Martin
  - Benchmark Consulting
  - Northrop Grumman
  - Elparazim
  - Fraunhofer Gesellschaft/Institute for Experimental Software Engineering (IESE)
- 

The following companies supported this specification:

- University of York
- Dalian University of Technology
- Universidad Carlos III de Madrid
- Carnegie Mellon University

### 6.3 How to Proceed

The rest of this document contains the technical content of this specification.

Clause 7. Specification overview - Provides design rationale for the SACM Argumentation Metamodel specification.

**Part 1** of the specification defines the normative common elements. This part includes three clauses. Material in this part of the specification is related to all compliance points.

Clause 8. SACM Base classes define the common base classes of the Structured Assurance Case Metamodel. Clause 9. SACM Packages define the common packages of the Structured Assurance Case Metamodel.

Clause 10. SACM Terminology defines the common terminology classes of the Structured Assurance Case Metamodel.

**Part 2** of the specification defines the SACM Argumentation metamodel. The Argumentation Metamodel defines the catalog of elements for constructing and interchanging structured statements describing argumentations. Material in this part of the specification is related to the Assurance Case and Argumentation compliance points, and is not required for the Evidence Container compliance point. This part includes a single clause. The non-normative Annex B contains some examples of the SACM XML interchange format for Argumentation, and describes how SACM Argumentation is related to existing graphical notations for describing structured arguments, such as the Goal Structuring Notation (GSN) and the Claims-Arguments-Evidence (CAE) notation.

Clause 11. The SACM Argumentation Metamodel - Provides the details of the Argumentation Metamodel specification.

**Part 3** of the specification defines the SACM Artifact Metamodel. The Artifact Metamodel defines the catalog of elements for constructing and interchanging precise statements involved in evidence-related



# 11 SACM Argumentation Metamodel

## 11.1 General

This chapter presents the normative specification for the SACM Argumentation Package. It begins with an overview of the metamodel structure followed by a description of each element.

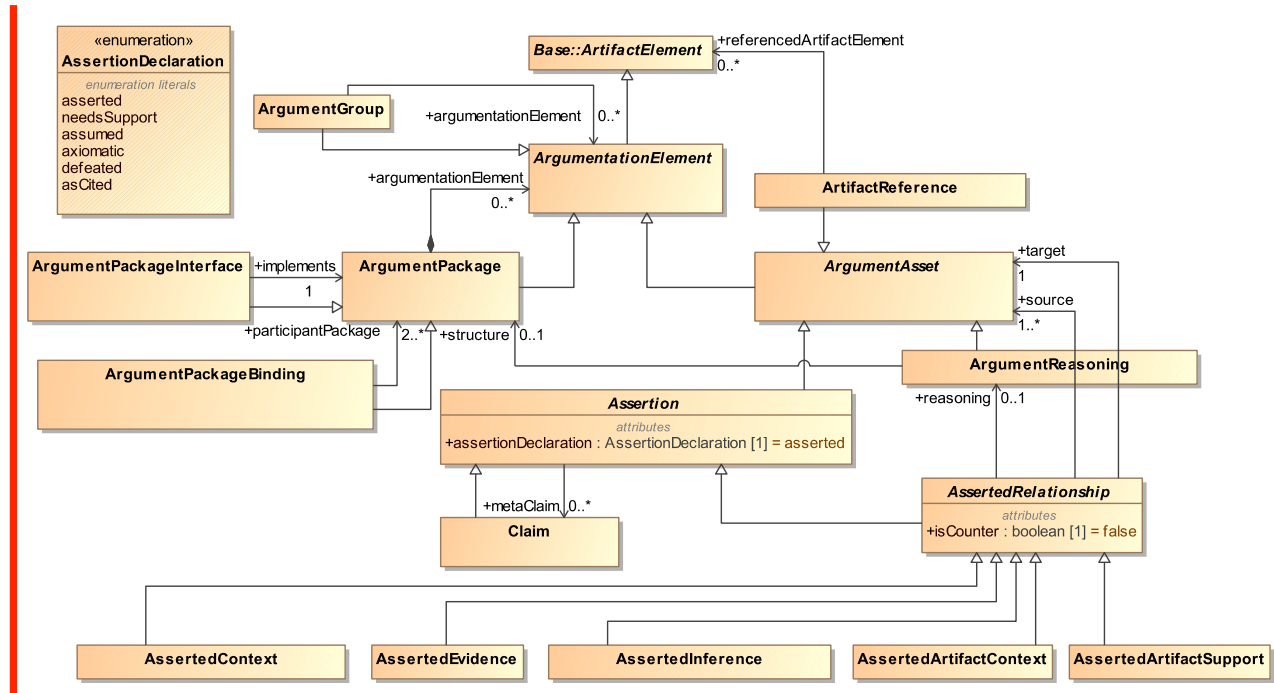


Figure 11.1 - Argumentation Package Diagram

This portion of the SACM model describes and defines the concepts required to model structured arguments. Arguments are represented in SACM through explicitly representing the Claims and citation of artifacts (e.g., as evidence) (ArtifactReference), and the ‘links’ between these elements – e.g., how one or more Claims are asserted to infer another Claim, or how one or more artifacts (referenced by ArtifactReference) are asserted as providing evidence for a Claim (AssertedEvidence). In addition to these core elements, in SACM it is possible to provide additional description of the ArgumentReasoning associated with inferential and evidential relationships, represent counter-arguments and counter-evidence (through isCounter:Boolean), and represent how artifacts provide the context in which arguments should be interpreted (through AssertedContext).

The packaging of structured arguments into ‘modular’ argument packages is enabled through ArgumentPackages. Users are able to declare interfaces for their packages through the use of ArgumentPackageInterface. Within an ArgumentPackageInterface, users create citations of the argumentation elements they select to disclose to external parties. Users are able to integrate ArgumentPackages through the use of ArgumentPackageBinding. An ArgumentPackageBinding binds ArgumentPackages together by including the declared ArgumentPackageInterfaces for the ArgumentPackages, it may contain additional argument structures to provide the rationale of the binding. It is also possible within a package to cite elements contained within other argument packages (through ArtifactReference).

# 12 Artifact Classes

## 12.1 General

This chapter presents the normative specification for the SACM Artifact Package. It begins with an overview of the metamodel structure followed by a description of each element.

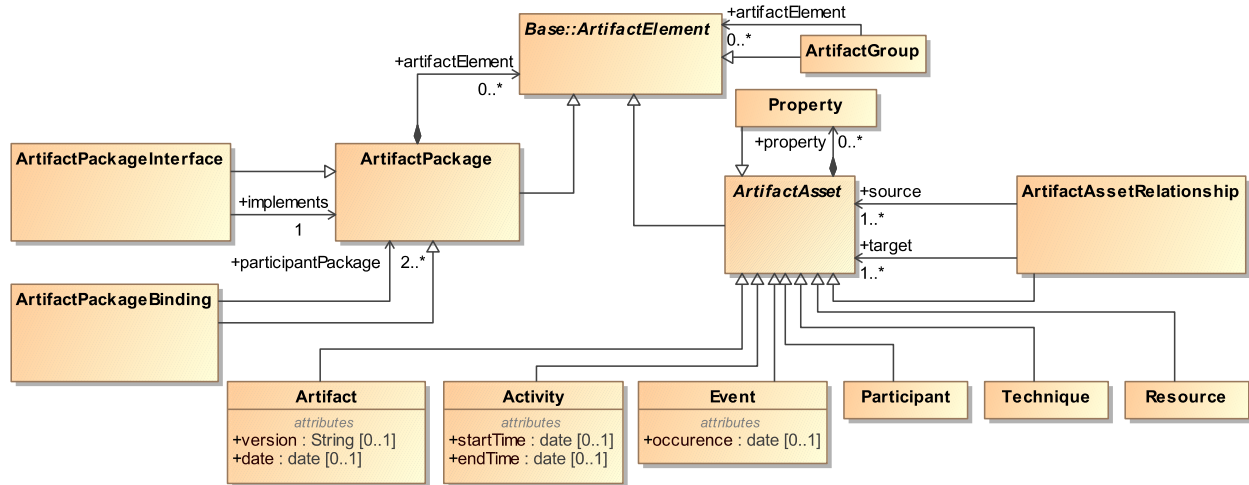


Figure 12.1 - Artifact Package Diagram

Artifacts correspond to the main evidentiary elements of an assurance case. By means of assertions (AssertedEvidence with isCounter = true/false), artifacts can be referenced (using ArtifactReferences) as supporting claims and arguments.

In general, artifacts are managed when the corresponding objects are available. For example, a test case is linked to the requirement that validates once the test case has already been created. However, artifact management might also require the specification of patterns (or templates) in order to allow a user, for instance, to indicate that a given artifact must be created but it has not yet. A common scenario of this situation corresponds to the process during which a supplier and a certifier have to agree upon the artifacts that the supplier will have to provide as assurance evidence for a system. As a result of this process, artifact patterns could be specified, and such patterns would need to be made concrete during the lifecycle of the system. Artifact patterns are specified by means of the attribute 'isAbstract' (SACMElement). For example, a supplier and a certifier might agree upon the need for maintaining a hazard log during a system's lifecycle. Such a hazard log would initially be modeled as an Artifact that is abstract. Once created, the value of this attribute of the hazard log would be 'false'. The specification of artifact patterns also facilitates their reuse, as the corresponding artifacts might have to be created in the scope of more than one assurance case effort. Using again hazard logs as an example, their structure might be the same for several systems, thus all the corresponding hazard logs might be based on a same abstract Artifact.

When made concrete, an Artifact can relate to many different types of information necessary for developing confidence in the Artifact and thus for assurance purposes. Such information can be regarded as meta-data or provenance information about an Artifact, provides information about its management, and is specified with the rest of specializations of ArtifactAsset. Using a design specification as an example, properties (Property) could be specified regarding its quality (completeness, consistency...), and it would have a lifecycle with events such as its creation and modifications. The specification could be created by using UML (Technique) in an Activity named 'Specify system design', stored in a Resource corresponding to a diagram created with some modeling tool, and later used as input for